6. In a high-performance computing application, you need to frequently calculate the square of numbers. To optimize these calculations, you decide to use an inline function.

#include <iostream>

// Inline function to calculate the square of a number

inline int square(int x) {

return x \* x;

}

int main() {

int num;

std::cout << "Enter a number: ";

std::cin >> num;

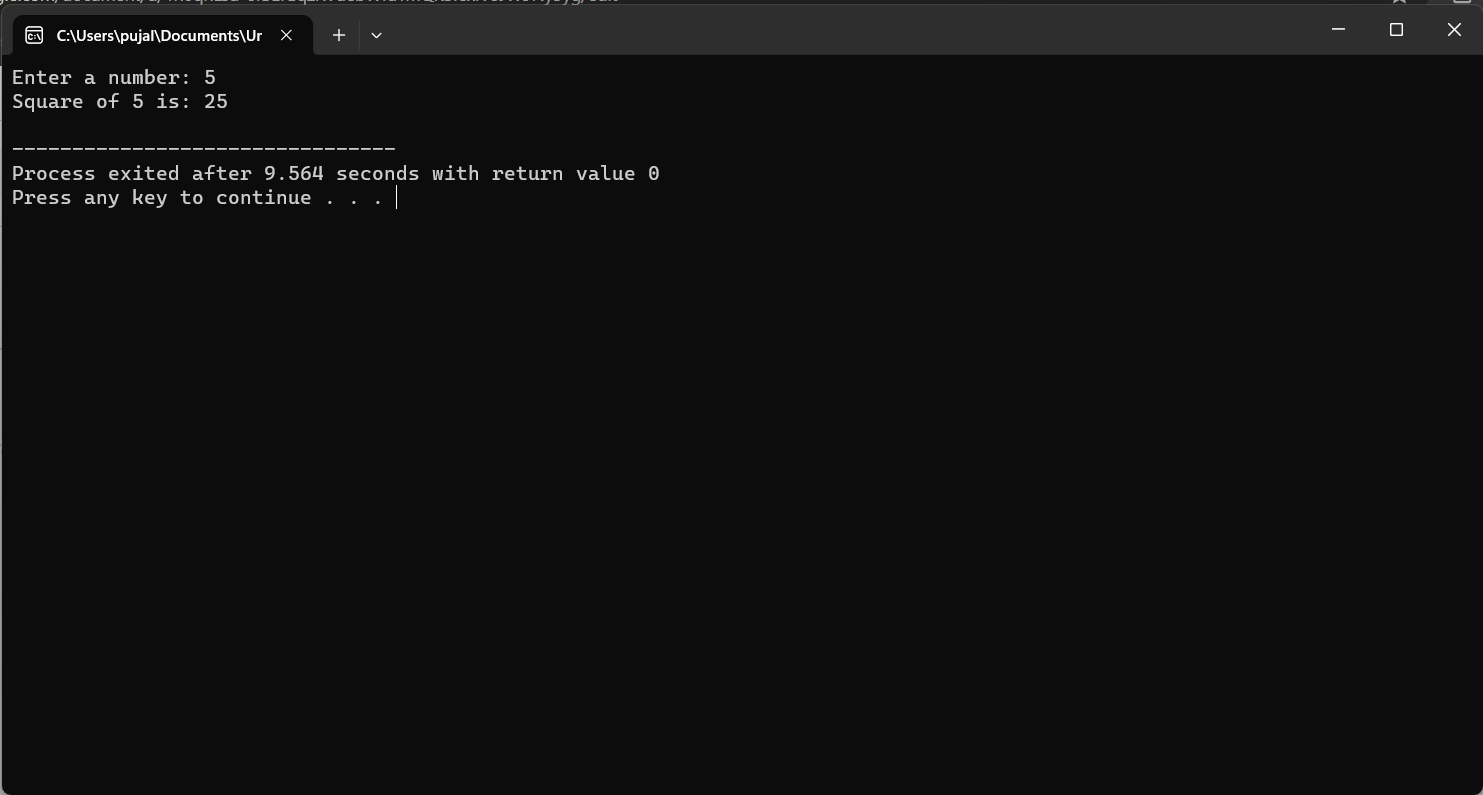
// Calculate the square using the inline function

int result = square(num);

std::cout << "Square of " << num << " is: " << result << std::endl;

return 0;

}



7.Design a configureGame function that includes default arguments, allowing users to specify only the settings they want to change from the defaults

#include <iostream>

#include <string>

using namespace std;

// Function to configure game settings with default arguments

void configureGame(int boardSize = 8, int numPlayers = 2, string playerName = "Player") {

cout << "Game Settings:" << endl;

cout << "Board Size: " << boardSize << endl;

cout << "Number of Players: " << numPlayers << endl;

cout << "Player Name: " << playerName << endl;

}

int main() {

// Use default settings

cout << "Default Game Settings:" << endl;

configureGame();

// Configure game with custom settings

cout << "\nCustom Game Settings:" << endl;

configureGame(10, 4, "John");

// Change only one setting

cout << "\nChange Board Size Only:" << endl;

configureGame(12);

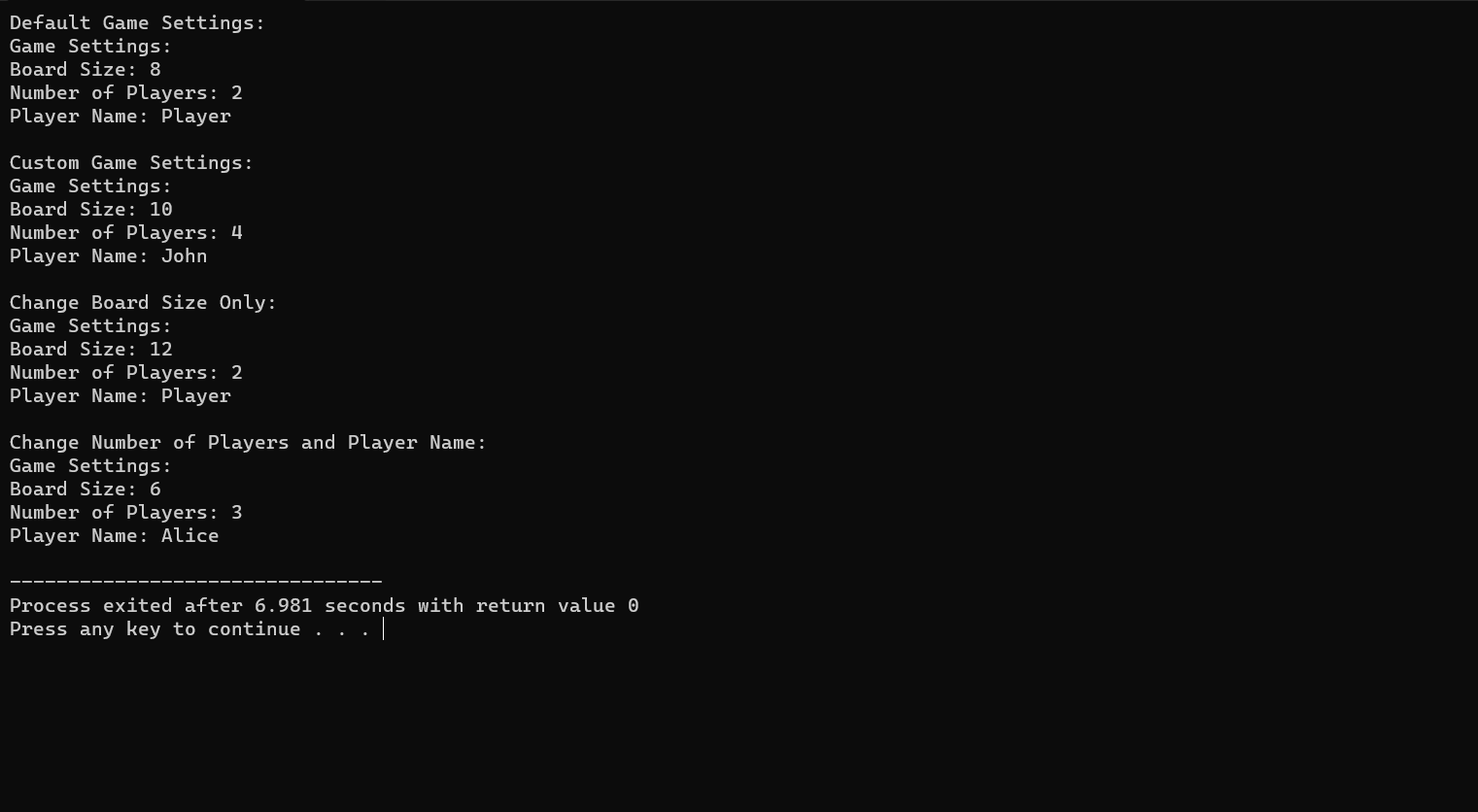
// Change only two settings

cout << "\nChange Number of Players and Player Name:" << endl;

configureGame(6, 3, "Alice");

return 0;

}



8. Calculate the average temperature of the month.

Find the maximum and minimum temperatures recorded in the month.

Determine the number of days the temperature was above the monthly average.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int daysInMonth;

cout << "Enter the number of days in the month: ";

cin >> daysInMonth;

// Read temperatures for each day

vector<int> temperatures(daysInMonth);

cout << "Enter temperatures for each day of the month:\n";

for (int i = 0; i < daysInMonth; ++i) {

cout << "Day " << i + 1 << ": ";

cin >> temperatures[i];

}

// Calculate the average temperature

double sum = 0;

for (int temp : temperatures) {

sum += temp;

}

double averageTemperature = sum / daysInMonth;

// Find the maximum and minimum temperatures

int maxTemperature = \*max\_element(temperatures.begin(), temperatures.end());

int minTemperature = \*min\_element(temperatures.begin(), temperatures.end());

// Determine the number of days above average

int daysAboveAverage = count\_if(temperatures.begin(), temperatures.end(),

[&](int temp) { return temp > averageTemperature; });

// Output results

cout << "\nAverage temperature of the month: " << averageTemperature << "°C\n";

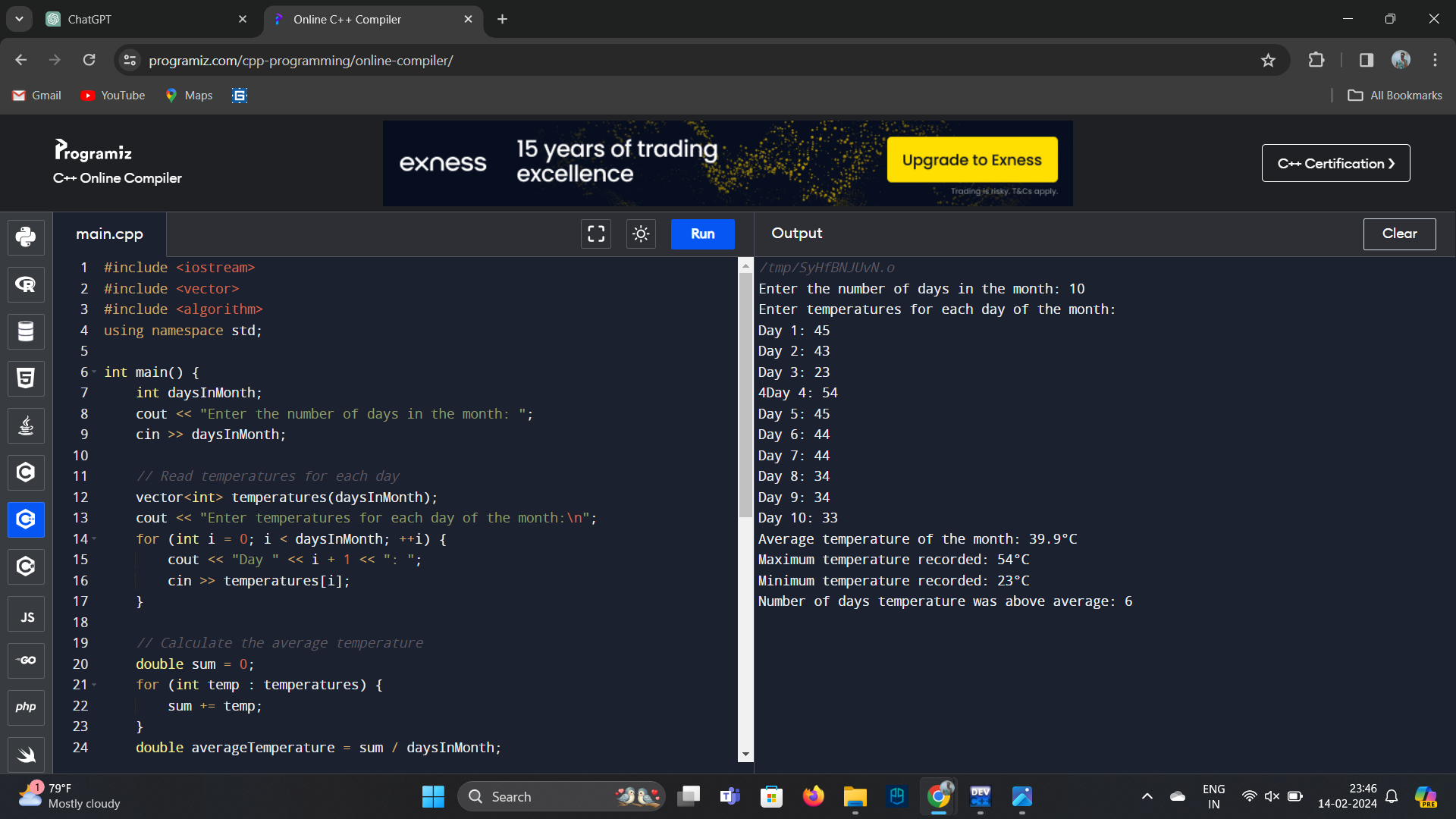
cout << "Maximum temperature recorded: " << maxTemperature << "°C\n";

cout << "Minimum temperature recorded: " << minTemperature << "°C\n";

cout << "Number of days temperature was above average: " << daysAboveAverage << endl;

return 0;

}



9.Problem Statement: Implement a simplified e-commerce cart system. You need to write functions to add an item to the cart, calculate the total cost of items in the cart including a fixed tax rate, and remove an item from the cart. Assume each item is represented by a struct containing a name, price, and quantity.

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

// Struct to represent an item in the cart

struct CartItem {

string name;

double price;

int quantity;

};

// Function to add an item to the cart

void addItemToCart(vector<CartItem>& cart, const string& name, double price, int quantity) {

cart.push\_back({name, price, quantity});

cout << "Item added to cart: " << name << endl;

}

// Function to calculate the total cost of items in the cart (including tax)

double calculateTotalCost(const vector<CartItem>& cart, double taxRate) {

double totalCost = 0;

for (const auto& item : cart) {

totalCost += item.price \* item.quantity;

}

totalCost \*= (1 + taxRate); // Apply tax

return totalCost;

}

// Function to remove an item from the cart

void removeItemFromCart(vector<CartItem>& cart, const string& name) {

auto it = remove\_if(cart.begin(), cart.end(),

[&](const CartItem& item) { return item.name == name; });

if (it != cart.end()) {

cart.erase(it, cart.end());

cout << "Item removed from cart: " << name << endl;

} else {

cout << "Item not found in cart: " << name << endl;

}

}

int main() {

vector<CartItem> cart;

// Add items to the cart

addItemToCart(cart, "Laptop", 1000.0, 1);

addItemToCart(cart, "Smartphone", 500.0, 2);

// Calculate total cost with 10% tax rate

double totalCost = calculateTotalCost(cart, 0.1);

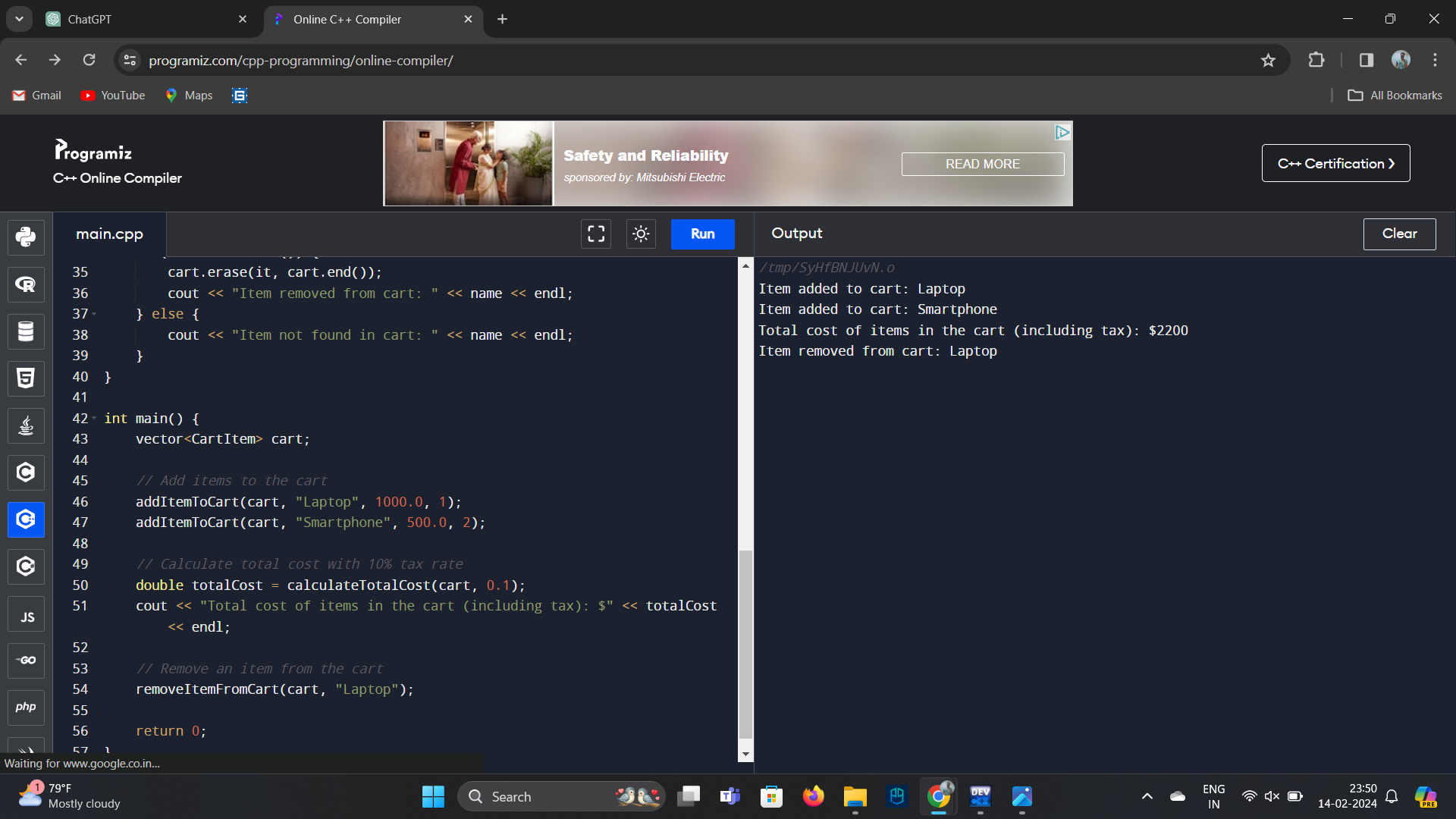
cout << "Total cost of items in the cart (including tax): $" << totalCost << endl;

// Remove an item from the cart

removeItemFromCart(cart, "Laptop");

return 0;

}



10. Matrix addition :

#include<iostream>

using namespace std;

int prime(int n ){

bool isprime=true;

for(int i=2;i<n;i++){

if(n%i==0){

cout<<n<<"is not a prime number:";

isprime=false;

break;

}

}

};

int main(){

int num;

cout<<"enter the given number:";

cin>>num;

prime(num);

if(isprime==true){

cout<<n<<"is a prime number";

}

}

